

**AD-A256 106**

Public reporting burden  
gathering and maintaining  
collection of information  
Davis Highway, Suite 2050  
Washington, DC 20503

**PAGE**

Form Approved  
OMB No. 0704-0188

per response, including the time for reviewing instructions, searching existing data sources, of information. Send comments regarding this burden estimate or any other aspect of this  
Headquarters Services Directorate for Information Operations and Reports, 1215 Jefferson  
and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

**1. AGENCY U****3. REPORT TYPE AND DATES COVERED**

FINAL - 1 Jul 89 - 30 Jun 92

**4. TITLE AND SUBTITLE**

"GOALS VERSUS ALGORITHMS" (U)

**5. FUNDING NUMBERS**

61102F

2304/A5

**6. AUTHOR(S)**

Dr. Peter J. Huber

AFOSR-TR-

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

Dept of Mathematics  
Massachusetts Institute of Technology  
77 Massachusetts Avenue  
Cambridge MA 02139

**8. PERFORMING ORGANIZATION REPORT NUMBER**

AFOSR-TR-

PHJ-91-2

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

AFOSR/NM  
Bldg 410  
Bolling AFB DC 20332-6448

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

AFOSR-89-0412

**11. SUPPLEMENTARY NOTES**

DTIC  
ELECTE  
OCT 7 1992  
S C D

**12a. DISTRIBUTION/AVAILABILITY STATEMENT**Approved for public release;  
Distribution unlimited**12b. DISTRIBUTION CODE**

UL

**13. ABSTRACT (Maximum 200 words)**

Computational methods in statistics often are defined through an algorithm. It is argued that a precise finite sample specification of the goals to be achieved by the algorithm is at least equally important. The issues are discussed in the special case of Projection Pursuit Regression. An interesting initial result of this work, which is still in progress, is that the Friedman-Stuetzle algorithm appears to be systematically biased toward overfitting.

**14. SUBJECT TERMS**

2200-18  
92-26577  
18  
136

**15. NUMBER OF PAGES**

12

**16. PRICE CODE****17. SECURITY CLASSIFICATION OF REPORT**

UNCLASSIFIED

**18. SECURITY CLASSIFICATION OF THIS PAGE**

UNCLASSIFIED

**19. SECURITY CLASSIFICATION OF ABSTRACT**

UNCLASSIFIED

**20. LIMITATION OF ABSTRACT**

SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)  
Prescribed by ANSI Std Z39-18  
298-102

# Goals versus Algorithms

Peter J. Huber\*  
Department of Mathematics  
Massachusetts Institute of Technology  
Cambridge, MA 02139, USA

September 27, 1991

Accepted	Not Yet	<input checked="" type="checkbox"/>
Not Yet	Accepted	<input type="checkbox"/>
Not Yet	Not Yet	<input type="checkbox"/>
Not Yet	Not Yet	<input type="checkbox"/>
* Information		
Probability Codes		
Basic and/or		
Dist	Special	<i>A-1</i>

## Abstract

Computational methods in statistics often are defined through an algorithm. It is argued that a precise finite sample specification of the goals to be achieved by the algorithm is at least equally important. The issues are discussed in the special case of Projection Pursuit Regression. An interesting initial result of this work (which still is in progress) is that the Friedman-Stuetzle algorithm appears to be systematically biased toward overfitting.

## 1 Introduction

Computer intensive methods in statistics nowadays are defined more often through specific algorithms than through specific goals. Names such as PPR (Projection *Pursuit* Regression) or ACE (*Alternating* Conditional Expectations), correspondingly refer to the process rather than to its end product.

In part, this is a matter of philosophy. Another, more practical reason is that precise finite sample goals are hard to specify, and on top of that, after one has specified them, one is confronted with an even harder task: one then is obliged to produce an algorithm that approaches those goals. With a mere algorithm, it suffices--at least for a start--to demonstrate that it works in a number of concrete examples.

Typically, simple goals exist in the abstract population case only. For example, such a goal might be to minimize the expectation of the average squared prediction error. It is difficult and ambiguous to translate such a description to a non-parametric finite sample situation, where the true underlying distribution is unknown, where smoothers must take over the role of conditional expectation operators, and where the bandwidth of such smoothers will have to be determined by cross-validation from the same sample itself. It may actually be easier and more straightforward to translate it by heuristic analogy directly into an iterative algorithm that makes sense in the finite sample case.

The custom of specifying an algorithm rather than its goals has the embarrassing (or should one say welcome?) side effect that alternative algorithms for the same problem are nearly impossible

\*Prepared with the partial support of Contract No. AFOSR-89-0412

to compare. If the algorithm fails in a specific instance, is the algorithm to blame or is it a fault of the problem (non-existence of a solution)? Without a formal specification of goals the question is vacuous.

I believe it is important to think about specific finite sample goals in order to shore up the algorithms. More precisely, those goals should help to more fully understand them, to establish firmer results about them, and, for example, allow us to assess the effects of compromises in favor of computational efficiency. This essay shall discuss some of the issues in the specific example of Projection Pursuit Regression (PPR). Its writing was triggered by some problems we ran into when trying to apply PPR to a specific low-noise fitting problem (see Section 3 below).

## 2 The Goals of PPR

In the population case, the goals of PPR can be described simply and accurately. Assume  $(\mathbf{X}, Y)$  is a random variable with values in  $\mathbf{R}^p \times \mathbf{R}$ . The task is to approximate the response surface

$$f(\mathbf{x}) = E(Y|\mathbf{X} = \mathbf{x}) \quad (2.1)$$

by a function representable as a sum of ridge functions:

$$g(\mathbf{x}) = \sum_1^m g_j(\mathbf{a}_j^T \mathbf{x}). \quad (2.2)$$

If  $m$  and the projection directions  $\mathbf{a}_1, \dots, \mathbf{a}_m$  are kept fixed, the best approximation in terms of the expected mean square error

$$E(Y - g(\mathbf{X}))^2 \quad (2.3)$$

is obtained by a function  $g$  uniquely characterized by the property that its conditional expectations agree with those of  $f$  in all  $m$  projection directions:

$$E(f(\mathbf{X})|\mathbf{a}_j^T \mathbf{X}) = E(g(\mathbf{X})|\mathbf{a}_j^T \mathbf{X}). \quad (2.4)$$

See Huber (1985, p. 466) and Huber (1991) for more precise results about the existence and uniqueness of the solution. Then one should optimize over all possible projection directions, and select the lowest possible number  $m$  of summands giving a satisfactory approximation. It is clear for *a priori* reasons that one cannot expect a unique optimal set of directions, because of the well-known polynomial indeterminacy of the representation (2.2).

We can rewrite the last displayed equation as

$$g_j(\mathbf{a}_j^T \mathbf{X}) = E(Y - g(\mathbf{X}) + g_j(\mathbf{a}_j^T \mathbf{X})|\mathbf{a}_j^T \mathbf{X}) \quad (2.5)$$

and turn it into an algorithm iteratively improving the component functions  $g_j$  by backfitting: assume the right hand side of this equation is based on the currently best versions of  $g$  and  $g_j$ , then the left hand side defines a new, improved version of the summand  $g_j$ .

The PPR algorithms proposed by Friedman and Stuetzle (1981) are very clever finite sample versions of such an iterative approximation, with compromises in favor of efficient computation. The basic idea is to replace conditional expectations  $E(\cdot|\mathbf{a}_j^T \mathbf{X})$  by scatterplot smoothers, and to build

up the sum of ridge functions by a greedy algorithm: find a first direction and a first summand capturing as much as possible of the variability, then apply the algorithm to the residuals in order to find a second direction and ridge function, and so on. The trickiest part is that the algorithm must be tuned indirectly, based on goodness of prediction rather than on goodness of fit, and hence must use cross-validation techniques within the optimization loop.

Friedman and Stuetzle noticed that after two or more summands have been found, the quality of the approximation can be improved by the finite sample version of backfitting, and often considerably so. That is, one leaves out a summand in the estimated fit

$$g(\mathbf{x}_i) = \sum_1^m g_j(\mathbf{a}_j^T \mathbf{x}_i), \quad (2.6)$$

say  $g_j$ , and redetermines a new  $g_j$  by smoothing the partial residuals

$$r_i^{-j} = \mathbf{y}_i - g(\mathbf{x}_i) + g_j(\mathbf{a}_j^T \mathbf{x}_i)$$

against the  $\mathbf{a}_j^T \mathbf{x}_i$ , repeatedly cycling through all values of  $j$ . We note that neither the projection direction  $\mathbf{a}_j$  nor the function  $g_j$  occur in the partial residual (the terms involving them cancel out), so one can use the opportunity to improve the direction at the same time. At least in principle one might iterate this procedure to convergence, but in practice one will prefer to stop much earlier.

As in the population version, one does aim for a best  $m$ -tuple of directions, minimizing the sum of the squared residuals  $r_i = y_i - g(\mathbf{x}_i)$ .

A comparison of this procedure with the population case raises several interesting mathematical questions. I believe they are unsolved if we except the special case to be considered in Section 5. Assume that the projection directions and the smoothers are kept fixed (i.e., no adjustment by cross-validation during the iterations). Does the finite sample backfitting process have a fixed point? Is it unique? What exactly, if anything, is minimized by this fixed point? Does backfitting converge to that fixed point?

### 3 A Case Study: Low-noise PPR

The application in question was to machine-learn a function from a sample of approximately 1,000 of its values (joint work with Ying Zhao and Chris Atkeson). In a specific example, the functions to be learned were the torques needed to move the endpoint of a two-joint robot arm along arbitrary paths in space-time. That is, the robot would have to learn two smooth functions connecting the six state variables (two angles and their first two derivatives) to the torques applied at the two joints, without knowing and making use of the underlying dynamics. In our examples, the independent variables were random, either uniformly or normally distributed, in up to 6 dimensions, while the dependent variables were for all practical purposes noise-free.

When we applied PPR to this problem we found, much to our surprise, that the fitted “smooth” function  $g$  tended to be a lot noisier than the original raw data, and we puzzled for months about the reason.

We quickly eliminated *Suspect No. 1: Programming error*, in favor of *Suspect No. 2: Numerically poor smoothing algorithms*. We had primarily been using local straight line fits and smoothing

splines. The computationally efficient updating algorithm for fitting local straight lines is known to be numerically poor, and the condition number of smoothing spline algorithms depends on the ratio between the range of the abscissae and the smallest interpoint distance, and thus, depending on the projection direction, can become arbitrarily poor.

After convincing ourselves that this was not the cause of our problem, we looked into the more esoteric pathologies of scatterplot smoothers, some of which are discussed in Buja, Hastie, and Tibshirani (1989). *Suspect No. 3* was *Instability through iteration*: some innocent looking linear smoothing operators have singular values slightly larger than 1, and may go astray after continued iterations. While this was not the cause of our problem, we unintentionally verified that it can be a serious issue when we checked *Suspect No. 4*, *Premature stopping*: a frequent problem with greedy algorithms (for example, with the steepest descent methods) is that they initially exhibit rapid improvements and then grind to a near halt at the bottom of a valley, still far from the ultimate limit point. Our *Suspects No. 5 and 6* were *Concavity* (an effect analogous to collinearity, but having to do with the superposition of non-linear functions), and the closely related possibility of *Non-closedness* of the space of ridge function (in which case the component functions  $g_j$  may go out of bounds while the sum  $g$  stays bounded) (see Huber [1991]). Then there was *No. 7, Initial overfitting*: the latter stages of PPR fitting might not have been able to counteract the after-effects of over-fitting in the initial stages. There may have been a few more ephemeral suspects.

Ultimately, after eliminating all other possibilities, we found that the real culprit was a rather mundane *Bias problem*. All the more usual smoothers (in particular local straight lines and cubic smoothing splines) are able to follow straight lines exactly, but they are biased when the response surface is curved. In any projection the resulting bias in  $g_j$  is small for the central part of the set of projected sample points  $\mathbf{a}_j^T \mathbf{x}_i$ , but near the more sparsely populated ends of the range, the bias can become rather large. For any other sufficiently different projection direction, the biases deriving from the  $j$ th projection behave like random noise with a few interspersed outliers (the latter caused the by large biases occurring near the end-points of the range). This problem became even more serious because we mistakenly had believed that robustness (or more precisely, outlier resistance) was unimportant in our case, where the longest-tailed distributions happened to be the uniform and the normal.

Incidentally, the best results (i.e., smoothest and closest fits) were obtained through a laborious backfitting process with smoothing splines, cycling through  $j$  while slowly lowering the Lagrange multiplier  $\lambda$  in

$$\sum_i (r_i^{-\ell} - g_j(\mathbf{a}_j^T \mathbf{x}_i))^2 + \lambda \sum_j \int (g_j^{(k)}(z))^2 dz$$

to (almost) 0. That is, we were getting close to an interpolating spline (with  $k = 2$  or 3). This process had to be controlled rather carefully: for small  $\lambda$ , especially if the abscissae are irregularly spaced, smoothing splines suddenly can become unstable.

The following example exhibits the bias problem in the purest possible form. The response surface was defined as

$$y = 1000x_1^2 - 1000x_2^2,$$

and  $n = 1000$  error-free points  $(x_{1i}, x_{2i}, y_i)$  were generated from a standard bivariate normal distribution for  $(x_1, x_2)$ . An approximate representation

$$g(\mathbf{x}_i) = g_1(x_{1i}) + g_2(x_{2i})$$

was determined by backfitting the  $g_j$  and iterated to convergence. Two different smoothers were used: (1) local straight lines, and (2) cubic smoothing splines. The smoothing parameters were adjusted so as to obtain an approximately optimal fit of  $g_1$  and  $g_2$  to the true functions  $1000x_1^2$  and  $-1000x_2^2$ , respectively. The results are shown in Figures 1 and 2.

Figure 1 here

Figure 1: Local straight line fitter. The errors in the fit are on the order of 1% of the  $y$ -values; note the large errors near the end points in the plots of  $(g_j - \text{true})$  against  $x_j$ .

Figure 2 here

Figure 2: Cubic smoothing splines. The results are qualitatively similar, but the scale of the errors is about 2 orders of magnitude smaller than for the local straight line fitter.

## 4 Specification of Goals Through Fixed-Point Properties?

The overall task is to find algorithms that not only can be implemented efficiently, but which are also amenable to theory (convergence proofs, etc.). This task can be approached either from the side of the algorithms or from the side of the theoretical goals; it may be advantageous to work from both sides.

It usually is straightforward to translate a fixed-point property into an algorithm and vice versa. One may say that any iterative algorithm has its own, built-in goals: its set of fixed points. This leaves open three mathematical problems: (i) to show that a fixed point exists; (ii) to establish its uniqueness (or lack of uniqueness); and (iii) to show that the algorithm converges to it. Apart from that there is a conceptual problem: what is the interpretation of the fixed points, are their features compatible with the ideas one had in mind when devising the algorithm?

Already proving the existence of a fixed point can be surprisingly difficult. In principle, almost anything can happen. A fixed point may exist but be unstable. The algorithm may show convergence, chaotic behavior, or divergence, and the behavior of the algorithm may be different for different starting points. One big problem is that with any of the more sophisticated statistical algorithms, such as PPR, a single loop of the iteration is very complex and consists of the application of several different, typically non-linear operators. Even a simple, heuristically appealing linear smoothing operator may on closer scrutiny turn out to have singular values larger than 1, and thus pose the danger of an exponential explosion on iteration. The usual tools for proving fixed-point or stability properties (such as Liapunov functions) practically are equivalent to an external characterization of the fixed point in terms of a minimum principle.

If a smoother is defined in terms of a minimum principle, several of the above-mentioned problems do not occur. For example, a linear smoother then automatically corresponds to a symmetric

operator with eigenvalues between 0 and 1. On the other hand, smoothers defined in terms of a minimum principle typically are more expensive to calculate.

## 5 Specification of Goals Through Minimum Properties?

A precise goal in terms of a minimum problem almost always helps with convergence proofs, but on the other hand it may not provide much help toward constructing an efficient algorithm.

A somewhat simple-minded finite sample specification of the target function

$$g(\mathbf{x}) = \sum_1^m g_j(\mathbf{a}_j^T \mathbf{x}) \quad (5.1)$$

of PPR in terms of a minimum problem involves two terms: a penalty for the fitting error or *misfit*  $y_i - g(\mathbf{x}_i)$  at the observed points, and a penalty for the roughness of the component ridge functions  $g_j$ . The conceptually simplest (although not necessarily best) roughness penalty is the integrated square of some higher order derivative, corresponding to smoothing splines. To fix the idea, we shall use it for the following arguments (and afterwards explain what aspects are unsatisfactory). The two penalties are combined with the help of Lagrange multipliers into an expression of the form

$$\sum_i (y_i - g(\mathbf{x}_i))^2 + \sum_j \lambda_j \int (g_j^{(k)}(z))^2 dz. \quad (5.2)$$

This is a clean, strictly convex (actually quadratic) minimum problem. If we write down the variational conditions for the minimum, it is immediately evident that this minimum is uniquely characterized by the property that it is a fixed point of the backfitting process. More precisely, those variational conditions are equivalent to requiring that solving the spline smoothing problem with abscissae  $\mathbf{a}_j^T \mathbf{x}_i$ , ordinates  $y_i - g(\mathbf{x}_i) + g_j(\mathbf{a}_j^T \mathbf{x}_i)$  and Lagrange multiplier  $\lambda_j$  yields  $g_j$ , and this for every  $j$ .

Moreover, each backfitting step decreases the value of (5.2), and given that the minimum is unique, it follows from a simple compactness argument that the process must converge to that unique minimum.

Evidently, this formalization can be generalized in several details; for example, by replacing the square in the first term of (5.2) by a more general convex function, involving some weight and scale parameters in order to achieve robustness, and by pulling the Lagrange multipliers  $\lambda_j$  inside the integral and making them depend on  $z$  in order to allow for different degrees of smoothness in different parts of the domain.

However, even with such modifications the formalization remains unsatisfactory in several respects. First, it is difficult to determine good *a priori* values for the Lagrange multipliers. To be realistic, the above formulation merely specifies the form of the functions  $g_j$  as being smoothing splines, with the  $\lambda_j$  remaining free parameters. Thus, while retaining the form of the functions, we must in practice determine those free parameters from the data, ideally by solving yet another minimum problem.

Second, the fitting error is a relatively meaningless quantity. One really is interested in the prediction error and one should determine  $g$  so that it minimizes the prediction error rather than the

fitting error. There are several possibilities for squeezing a surrogate prediction error out of the fitting procedure, all called by the generic name “cross-validation.” But there is a fundamental difficulty: the prediction error is not experimentally observable unless we have additional data not used for determining the fit. Strictly speaking, this catches us in an impossible dilemma: if we try to determine the fit such that it minimizes the prediction error, we must use data we are forbidden to use! Because of the dilemma just mentioned, some caution is indicated when cross-validation is used inside an iterative process.

One possibility for cross-validation is to leave out single observations, one at a time, to re-determine the fit, and then to compare the results at the point where the observation has been left out. In general, this is a computationally expensive process ( $O(n^2)$  operations, although there may be shortcuts), and moreover it is only indirectly relevant to the original fitting procedure: if one leaves out an observation, one no longer uses the identical procedure; the carrier  $\{\mathbf{x}_i\}$  has been changed, and even the sample size is different. We therefore propose to use a version based on the fitting procedure itself, with the same sample size  $n$ , to be called *self-consistent prediction*. Often, it leads to the identical results as the leave-one-out approach, giving the latter an *a posteriori* justification.

Assume  $g$  is the fitted function. Its value at a point  $\mathbf{x} \in \mathbb{R}^p$  depends on the observed data  $(\mathbf{x}_i, y_i)$ ,  $i = 1, \dots, n$  and on some parameter(s)  $\lambda$  (for example, the Lagrange multiplier in the case of splines, or the bandwidth on the smoother):

$$g(\mathbf{x}) = g(\mathbf{x}; \mathbf{x}_1, \dots, \mathbf{x}_n, y_1, \dots, y_n, \lambda). \quad (5.3)$$

We define the *self-consistent predictor*  $\hat{y}_i$  of  $y_i$  at  $\mathbf{x}_i$  by the property that

$$\hat{y}_i = g(\mathbf{x}_i; \mathbf{x}_1, \dots, \mathbf{x}_n, y_1, \dots, y_{i-1}, \hat{y}_i, y_{i+1}, \dots, y_n, \lambda). \quad (5.4)$$

**Example: Linear fits.** Assume the fitted value is linear in the  $y_i$ :

$$\hat{y}_i = g(\mathbf{x}_i) = \sum_l h_{il} y_l \quad (5.5)$$

where the coefficients  $h_{il}$  do not depend on  $(y_1, \dots, y_n)$ , but they may depend in any way on the other data. Then the self-consistent predictor is defined by

$$\hat{y}_i = h_{ii} \hat{y}_i + \sum_{l \neq i} h_{il} y_l,$$

and it is easy to verify that the fitting error and the residual of the self-consistent predictor are related by

$$y_i - \hat{y}_i = (1 - h_{ii})(y_i - \hat{y}_i). \quad (5.6)$$

For later use we note the following consequence of this relation:

$$\hat{y}_i - \hat{y}_i = \frac{h_{ii}}{1 - h_{ii}}(y_i - \hat{y}_i). \quad (5.7)$$

Note that  $h_{ii}$  is the self-influence on the  $i$ th observation on its own fitted value, that is the change induced in the latter by a unit change in the former. The reciprocal of the self-influence  $h_{ii}$  might be taken as the formal definition of the effective bandwidth (or, perhaps more accurately, of the effective number of degrees of freedom) of the smoother at this point.

It is heuristically plausible that for “reasonable” non-linear fitting procedures a similar, approximately linear relation will hold; in specific cases it may be possible to derive the exact relationship.

Therefore, the misfit, if suitably scaled, can be used as a surrogate for the prediction error. For the moment, we shall continue the argument for linear smoothers.

In the backfitting process the smoothers are applied separately to the separate projections, with possibly different smoothing parameters  $\lambda_j$ . It is relatively straightforward to determine the coefficients of self-influence  $h_{j,ii}$  for the individual univariate backfitting smoothers.

However, we need to determine the overall self-influence  $d_{ii}$  of  $y_i$  on  $g(\mathbf{x}_i)$ . This is much more difficult; we are going to give an argument that the approximate relationship between the two is given by

$$\frac{d_{ii}}{1 - d_{ii}} \approx \sum_j \frac{h_{j,ii}}{1 - h_{j,ii}}. \quad (5.8)$$

A rigorous proof is still outstanding, but I believe the formula is asymptotically accurate for small self-influences. The empirical evidence suggests that it is too optimistic (i.e., the actual  $d_{ii}$  tend to be larger).

We shall write for short  $g(i) = g(\mathbf{x}_i)$  and  $g_j(i) = g_j(\mathbf{a}_j^T \mathbf{x}_i)$ ; a minus sign as in  $g_j(-i)$  shall denote the value at the  $i$ th point computed by self-consistent prediction leaving out observation  $y_i$ .

The solution  $g$  is a fixed point of backfitting: smoothing  $y_i - g(i) + g_j(i)$  restores  $g_j(i)$ . If at this fixed point we do a corresponding cross-validating smooth, omitting the  $i$ th observation, equation (5.6) gives

$$[y_i - g(i) + g_j(i)] - g_j(i) = (1 - h_{j,ii}) \{ [y_i - g(i) + g_j(i)] - g_j(-i) \} \quad (5.9)$$

or, with  $r_i = y_i - g(i)$  from (5.7):

$$g_j(i) - g_j(-i) = \frac{h_{j,ii}}{1 - h_{j,ii}} r_i. \quad (5.10)$$

If the self-influences  $h_{j,ii}$  are small, the changes will superimpose linearly. If we sum (5.10) over  $j$  and take (5.7) into account, we obtain approximately

$$g(i) - g(-i) = \frac{d_{ii}}{1 - d_{ii}} r_i = \sum_j \left( \frac{h_{j,ii}}{1 - h_{j,ii}} \right) r_i \quad (5.11)$$

which is the relation announced in (5.8).

Thus, if we want to minimize the cross-validated prediction error, we should minimize

$$\sum_i \left( \frac{y_i - g(\mathbf{x}_i)}{1 - d_{ii}} \right)^2 \quad (5.12)$$

by varying the parameters  $\lambda_j$  in (5.2). Note that for fixed smoothing parameters  $\lambda_j$  the  $d_{ii}$  are fixed.

For the sake of reducing the technical complications, one might prefer to replace the value  $d_{ii}$  in (5.12) by the average of  $d_{ii}$  over  $i$  (sometimes called “generalized cross-validation”). It may actually be preferable to average the scores  $\eta_{ii} = d_{ii}/(1 - d_{ii})$  rather than the  $d_{ii}$  themselves. At the same time, such averaging seems to improve stability (robustness) of the procedure.

More generally, taking standard robustness considerations into account, one might replace (5.12) by an expression of the form

$$\sum_i w_i \rho \left( \frac{y_i - g(\mathbf{x}_i)}{s_i(1 - d_{ii})} \right), \quad (5.13)$$

where  $\rho$  is a convex symmetric function, while  $s_i$  and  $w_i$  are weight and scale factors whose precise specification at the moment must be left open.

Friedman and Stuetzle used local straight line smoothers rather than splines, there were quick and dirty computational shortcuts (instead of an expensive full minimization, they would take the best out of three values), and there clearly was no intention to iterate the backfitting process to death. But essentially they applied cross-validation to their backfitting smoothers. In our case and terminology this amounts to adjusting the smoothing parameters  $\lambda_j$  by minimizing

$$\sum_i \left( \frac{y_i - g(\mathbf{x}_i) + g_j(\mathbf{a}_j^T \mathbf{x}_i) - g_j^{new}(\mathbf{a}_j^T \mathbf{x}_i)}{1 - h_{j,ii}} \right)^2 \simeq \sum_i \left( \frac{y_i - g(\mathbf{x}_i)}{1 - h_{j,ii}} \right)^2 \quad (5.14)$$

separately for each  $j$  during the backfitting process. It is not clear whether such a procedure converges at all. But it is evident from (5.8) that the self-influences used by this procedure are systematically too small (on average by a factor comparable to the number  $m$  of ridge functions), and thus the result will be biased towards overfitting. It remains to be seen how serious this bias can be.

Above, to fix the idea, we had used smoothing splines. This does not mean that I advocate to use them for PPR. Despite their conceptual elegance, smoothing splines have some unpleasant properties. They are expensive to compute, and if the abscissae are irregularly spaced, they tend to become unstable (not only numerically). The tentative recommendation is to use a variant of smoothing splines with a smaller number of fixed, more evenly spaced knots. The search for other good smoothers that can be characterized in terms of a manageable minimum principle is wide open.

I shall briefly mention an unsatisfactory aspect of a rather general nature. One should not have to worry about smoothness explicitly: at least in principle, the prediction error should be all that is needed. Any roughness penalty is based on the unwarranted assumption that the best predictors are furnished by some sort of low-pass filters. That is, high-frequency fluctuations are regarded as noise to be filtered out, while low-frequency fluctuations are to be modeled as structure. In defense of smoothness in the context of PPR, one might however adduce that structure with a high spatial frequency is practically impossible to find, and it manifests itself only in an extremely narrow angular range of projection directions.

## 6 References

Friedman, J.H. and Stuetzle, W. (1981). Projection Pursuit Regression. *J. Amer. Statist. Assoc.* **76** 817-823.

Huber, P.J. (1985). Projection Pursuit (with discussion). *Annals of Statistics* **13** 435-525.

Huber, P.J. (1991). Closedness of Spaces of Ridge Functions. *Tech. Report.* M.I.T., Dept. of Mathematics.

Buja, A., Hastie, T. and Tibshirani, R. (1989). Linear Smoothers and Additive Models. *Annals of Statistics*, 17 No. 2 453-555.



